THE Exidy Monitor

Volume I Issue 4 $1.00

Z-80 Part II
MEMTEST Explained
New Policies
User Group Volumes

Michael Vachon

THE EXIDY MONITOR

printed monthly

John T. Vachon, Chief Editor
Thomas F. Harper, Software Consultant
Bruce R. McGlothlin, Coordinator

Cover art by Michael J. Vachon

"The Exidy Monitor"
is sponsored by

COMPUTER MART OF MASSACHUSETTS
1395 MAIN STREET
WALTHAM, MA 02154
(617) 899-4540

THE EXIDY MONITOR
Volume I
Issue IV

As you can probably tell by our new newsletter, we're going through drastic changes here at the head-quarters for The Exidy Monitor. Our new newsletters will be printed up by a professional printer. This will mean that we will have more time to prepare and coordinate the newsletter which should make it much better than it has been in the past. Also, because we're having them printed up, we now can have copies of the newsletter that will be selling in the store for $1.00 each. This means that the $5.00 the User's Group member pays saves him $7.00 off the $12.00 per year book-shelf price. Back-orders are also being printed up and will be available for $1.00 each.

Our software library is being expanded and changed. These tapes will be professionally copied in large quantities so that we will always have at least ten copies of each volume in stock. Documentation will accompany each volume. See page 4 for a complete description of the volumes and their contents. Tapes will be available for $6.50 each for members and $9.99 each for non-members. Use the enclosed order blank to order tapes.

Tom Harper returns with the second part of his Z-80 primer. He will be discussing the differences between assembly language and machine language, the formatting of the assembly file; the conversion of assembly language to machine language; and he will begin to explore the basic instructions and bit manipulation of Assembly language programming.

We're pleased to see so many people responding to our ad in INTERFACE AGE (and hope that just as many will respond to our ad in PERSONAL COMPUTING). Our roster has increased to over 70 names, and we receive more requests every day. With this much interest, we feel that we must be doing something right! To keep this interest, we will continue to produce the best newsletter we can.

I'd also like to mention something that we mentioned in the last newsletter, and that is simply that

WE NEED PROGRAMS!!!

Finally, I'd like to request a favor from you. We would desperately like articles written by group members that tell about experiences they have had with their Sorcerer, what they are using it for, or what problems you have had. We want anything that will be informative or generally interesting. Please contact us if you wish to participate. We would also appreciate any letters or comments addressed to our "Ask LP" column.

Sorcerer User's Group Software Library Volumes

The Software library is still expanding for the Sorcerer computer. Presently the library has eleven volumes that will be discussed in detail below. We have ideas for more utilities and games that will become the contents of future volumes, but we still need more. Remember that this library should be comprised of programs that you, the user, has written.

The eleven volumes of programs that we currently have available contain mostly games. The volumes and what they contain are:

[VOL I]	AMAZE	draws mazes
	LUNAR	lunar lander simulator
	ROTE	letter-ordering game
[VOL II]	LUCAS	sets up "Lucas' Puzzle"
	RSP	game of Rock, Scissors, Paper
	CHECK	solitary checkers game
[VOL III]	DEPTH	depth-charge game where you must sink the TRS-80
	REVRS	number reversing game
	HIQUE	plays the ancient game of HI-Q
[VOL IV]	BIO	gives a bio-rhythm chart
	MIDAG	lets you control a middle-ages mannor
	MANDL	prints up a mandala on the Sorcerer
[VOL V]	HANGM	hangman program (173 difficult words)
	HPAWN	hexapawn
[VOL VI]	HANG2	32K version of HANGM with 600 words and 50 phrases
	HANG3	16K version of HANGM with 333 words and 50 phrases
[VOL VII]	WARPX	Start Trek game -- uses music board
	TREK1	Instructions
[VOL VIII]	WARPZ	Start Trek game -- uses music box
	TREK1	Instructions
[VOL IX]	BANER	prints baners
	BOXOP	experimental math trial and error demo
[VOL X]	DEMO	Exidy 32K demo
	CHAR	Exidy Character generator program
[VOL XI]	DUMBT	dumb terminal program
	DUMB2	dumb terminal program for IBM and DEC

Machine Language Primer

Part II

by Tom Harper

Welcome to this months edition of the Z-80 machine language primer. This month we will overview the process involved in making a machine language program, as well as familiarizing you, the beginning machine language programmer, with some of the basic operations your powerful Z-80 CPU can perform.

The process of creating an assembly language program, both in structure and meaning, is a very complicated one to present to the average person who seldom steps from the cuddling arms of BASIC. However, I feel that the readers of this article are of a generally higher intellectual level than most, so I will present it rather straightforwardly.

An assembly language program is a symbolic representation of it's machine executable version, the machine language program. The assembly language program is in no way executable by the Z-80 itself, but must first be translated into the machine readable machine language program by what is known as an assembler. Different assemblers have different characteristics and features, much like different versions of BASIC. Some allow what is known as MACROS, which are somewhat like defined functions in BASIC, and simply allow repetitive reiterations of the same lines of code. Some allow you to use arithmetic expressions instead of numbers like '2*3+10/2' instead of plain old '11'. As a whole, however, most assemblers are very close to each other in features. One of the most problematical differences is the assemblers authors choice of mnemonics (the 'm' is silent). Mnemonics, as any dictionary freak will tell you, are symbols which assist in remembering normally hard to remember items. In this case, we are talking about multiple character names which are assigned to represent each of the Z-80 instructions, and are in turn converted by the assembler into the machine runable op-codes(operation codes!). Another valuble feature of an assembler is the ability to use LABELS. Labels are much like line numbers in BASIC and allow the user places to GOTO. The big item here is that since you are working with raw memory locations instead of lines of code, when you GOTO, you must give an absolute memory location. Now, suppose you forgot an instruction...where does it go? You put it where it belongs and your entire program must be moved down, screwing up all your

GOTO´s. Graphically, picture the following BASIC program:

```
     1 PRINT 10/2*3
     2 X=1
     3 PRINT X,X*2
     4 IF X<10 THEN GOTO 2
     5 PRINT "ALL DONE"
     6 GOTO 9
     7 PRINT "THIS WILL NEVER BE RUN"
     8 REM THIS NEITHER
     9 END
```

Now, suppose you wanted to put in another line between lines 1 and 2...you´d have to retype everything all over again! Of course, this is bypassed by numbering our programs by 10´s, but to leave 10 empty memory locations in between each instruction is unbelievably wasteful of memory. Just to put your mind to rest, BASIC is special with line numbers, so you can number them anyway you want without taking up extra memory. As a sample, I have included below a short assembly language program which I will explain which will hopefully make everything clear as far as strucure and effect of assembly language programming:

```
;------------------------------------------------
;program to help explain assembly language
;written by Tom Harper, CMI
;revision level one - 5/15/79
;
XOR     A       ;CLEAR ACCUMULATOR
ADD     A,1     ;ADD A ONE TO THE ACCUMULATOR
ADD     A,2     ;ADD A TWO TO THE ACCUMULATOR
ADD     A,3     ;ADD A THREE TO THE ACCUMULATOR
ADD     A,4     ;ADD A FOUR TO THE ACCUMULATOR
ADD     A,5     ;ADD A FOUR TO THE ACCUMULATOR
ADD     A,6     ;ET CETERA...
ADD     A,7
ADD     A,8
ADD     A,9
ADD     A,10
```

```
;-----------------------------------------------
```

What this program does is add ten numbers together and store it in the accumulator. For those of you who don't know what an accumulator is, it is one of the registers refered to in the last part of the MLP. The accumulator handles most of the arithmetic operations like adding, subtracting, etc. The stuff following the semi-colons(;) are comments which are similar to remarks in BASIC and allow the programmer to remember what he was doing. The first line, which includes the mnemonic 'XOR A' is utilizing one of the basic operations of the CPU, which are listed below:

AND: takes two binary numbers and copies the common bits as a result like this:

```
         10100110
    AND  11110011
         --------
      =  10100010
```

OR: takes two binary numbers and copies a one bit if either of the other bits are one:

```
         10100110
     OR  11110011
         --------
      =  11110111    <-- note the single zero because
                         the two bits over it are both
                         zero.
```

XOR: takes two numbers and copies a one bit only if one bit or the other is on but not both:

```
         10100110
    XOR  11110011
         --------
      =  01010101
```

NOT: takes a single number and flips all the bits:

```
         NOT 10101010 = 01010101
         NOT 11110000 = 00001111
```

Thus, our XOR A instruction exclusive or's the accumulator with itself, zeroing all the bits !!!!!

For instance, say the accumulator contained the binary number 10000001, and we XOR'ed it with itself:

```
        10000001
    XOR 10000001
        --------
        00000000     <---- Zero!!
```

The next instructions 'ADD' 1 to the accumulator, then add 2 to the accumulator, etc.

A much simpler, shorter, more efficient version which also shows the use of labels follows:

```
;simpler program to add numbers from 1 to 10
;written by Tom Harper CMI
;---------------------------------------------
        XOR     A       ;CLEAR ACCUMULATOR
        LD      B,10    ;SET COUNT TO 10
LOOP:   ADD     A,B     ;ADD NEXT NUMBER
        DEC     B       ;DECREMENT B
        JMP     NZ,LOOP ;JUMP IF NUMBER <> 0
        HALT            ;HALT - DIE MACHINE!
;---------------------------------------------
```

This program functions exactly like the previous one, except that it is obviously better. We introduce some new mnemonics. These are 'LD' for load, which loads the B register with the value 10. Next we add the value in register B to the accumulator. Next the DEC instruction will decrement the B register (cut it to 9 then 8, etc.). Next is a JMP instruction. This functions much like the GOTO instruction. In this case, it is a conditional jump, the condition being *Not Zero* or NZ. Note that providing the operation of DECrementing the B register was not zero, the machine will jump to the label 'LOOP'. Now, as an example of labeling, if we forgot the DEC instruction, we could very simply insert it right in, with the assembler taking care of where LOOP really is. HALT just tells the CPU to stop execution, effectively killing the machine.

For anyone who has difficulty with some of this information, feel free to call me at 899-4540 for some more extensive explanation.

Next Issue, I'll go more into the instruction set of the Z-80, as well as tying up our description of how assembly language is formatted, including how the assembler formats our machine code file.

Explanation of MEMTEST

by Tom Harper

We've had some feedback concerning my ramtest program in volume two. As I was out for a few weeks with the flu, I was unable to present any documentation for the program and, in fact, the listing is incomplete and the program will not run as printed. The main problem is a quirk of my assembler which I use on a giant Z-80 system to prepare code for the Sorcerer, which is then downloaded. The problem is that the hex code for ASCII messages is not fully printed out, so the messages starting at locations '55', '5C', and '64' are incomplete, which causes the program to bomb out.

The completed code is thus:

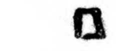
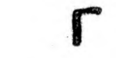
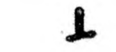
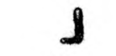
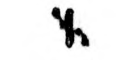
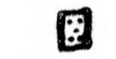
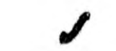
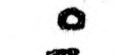
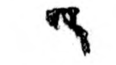
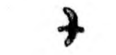
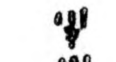
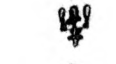
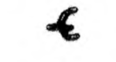
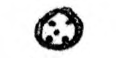
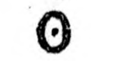
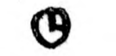
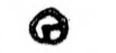
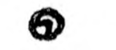
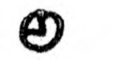
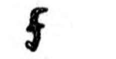
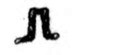
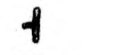
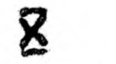
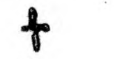
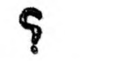
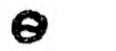
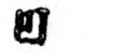
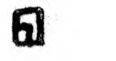
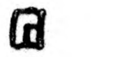
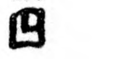
```
@Loc 55: 45 52 52 4F 52 20 00
@Loc 5C: 50 41 53 53 20 23 20 00
@Loc 64: 20 4E 4F 20 45 52 52 4F52 53 0D 0A 00
```

Now, for those of you who really don't understand what that sheet of slop I wrote means or does, this month's machine language primer includes an explanation on how assembly language files are structured, as well as what the code means. For those of you who are familiar with A.L., I will provide some info so you can play with the program. At the beginning of the program, there are several EQUate statements. These are CHOUT, BUFOUT, ADDOUT, HEXSP, and EPAGE. CHOUT is the Sorcerer character output routine, BUFOUT is the message output routine, ADDOUT is the sixteen-bit address output routine, HEXSP outputs an eight-bit hex number preceded by a space, EPAGE is the page number at which the program stops. In the listing you see, the page number is '20', which means that the program will stop testing at location 2000, or 8K. The variable START is set to the bottom of the program, this is where the test starts. To change the start and end locations, change all sixteen-bit references to START to your starting address, and all eight-bit references to EPAGE to the ending page (left two digits of the hex ending address.)

Well, have fun with it and good luck!!!
I'll take any calls pertaining to memtest personally.

Contributed by D. B. Trussell

SORCERER CHARACTER DESCRIPTION

Memory Location	HEX	Decimal	Character
F 800	00	00	
08	01	1	
F 810	02	2	
18	03	3	
F 820	04	4	
28	05	5	
F 830	06	6	
38	07	7	
F 840	08	8	
48	09	9	
F 850	0A	10	
58	0B	11	
F 860	0C	12	
68	0D	13	
F 870	0E	14	
78	0F	15	
F 880	10	16	
88	11	17	
F 890	12	18	
98	13	19	
F 8A0	14	20	
A8	15	21	
F 8B0	16	22	
B8	17	23	
F 8C0	18	24	
C8	19	25	
F 8D0	1A	26	
D8	1B	27	
F 8E0	1C	28	
E8	1D	29	
F 8F0	1E	30	
F8	1F	31	

Memory Location	HEX	Decimal	Character
F 900	20	32	Blank
8	21	33	!
F 910	22	34	"
8	23	35	#
F 920	24	36	$
8	25	37	%
F 930	26	38	&
8	27	39	'
F 940	28	40	(
8	29	41	)
F 950	2A	42	*
8	2B	43	+
F 960	2C	44	,
8	2D	45	-
F 970	2E	46	.
8	2F	47	/
F 980	30	48	0
8	31	49	1
F 990	32	50	2
8	33	51	3
F 9A0	34	52	4
8	35	53	5
F 9B0	36	54	6
8	37	55	7
F 9C0	38	56	8
8	39	57	9
F 9D0	3A	58	:
8	3B	59	;
F 9E0	3C	60	<
8	3D	61	=
F 9F0	3E	62	>
8	3F	63	?

Memory Location	HEX	Decimal	Character
F A00	40	64	@
8	41	65	A
F A10	42	66	B
8	43	67	C
F A20	44	68	D
8	45	69	E
F A30	46	70	F
8	47	71	G
F A40	48	72	H
8	49	73	I
F A50	4A	74	J
8	4B	75	K
F A60	4C	76	L
8	4D	77	M
F A70	4E	78	N
8	4F	79	O
F A80	50	80	P
8	51	81	Q
F A90	52	82	R
8	53	83	S
F AA0	54	84	T
8	55	85	U
F AB0	56	86	V
8	57	87	W
F AC0	58	88	X
8	59	89	Y
F AD0	5A	90	Z
8	5B	91	[
F AE0	5C	92	\
8	5D	93	]
F AF0	5E	94	^
8	5F	95	_

Memory Location	HEX	Decimal	Character	Memory Location	HEX	Decimal	Character	Memory Location	HEX	Decimal	Character
F B00	60	96	'	F C00	80	128	|	F D00	A0	160	▫
8	1	97	a	8	81	129	|	8	A1	161	▫
F B10	2	98	b	F C10	82	130	|	F D10	A2	162	‖
8	3	99	c	8	83	131	|	8	A3	163	♦
F B20	4	100	d	F C20	84	132	‡	F D20	A4	164	♣
8	5	101	e	8	85	133	|	8	A5	165	♧
F B30	6	102	f	F C30	86	134	|	F D30	A6	166	♤
8	7	103	g	8	87	135	|	8	A7	167	▯
F B40	8	104	h	F C40	88	136	⊙	F D40	A8	168	▯
8	9	105	i	8	89	137	–	8	A9	169	▭
F B50	A	106	j	F C50	8A	138	–	F D50	AA	170	▱
8	B	107	k	8	8B	139	–	8	AB	171	/
F B60	C	108	l	F C60	8C	140	–	F D60	AC	172	\
8	D	109	m	8	8D	141	π	8	AD	173	+
F B70	E	110	n	F C70	8E	142	X	F D70	AE	174	–
8	F	111	o	8	8F	143	╭	8	AF	175	—
F B80	70	112	p	F C80	90	144	╮	F D80	B-	176	~
8	1	113	q	8	91	145	┌	8	B1	177	▒
F B90	2	114	r	F C90	92	146	–	F D90	B2	178	⊥
8	3	115	s	8	93	147	◸	8	B3	179	═
F BA0	4	116	t	F CA0	94	148	◹	F DA0	B4	180	├
8	5	117	u	8	95	149	▫	8	B5	181	▒
F BB0	6	118	v	F CB0	96	150	▫	F DB0	B6	182	‖
8	7	119	w	8	97	151	–	8	B7	183	‖
F BC0	8	120	x	F CD0	98	152	♠	F DC0	B8	184	░
8	9	121	y	8	99	153	♥	8	B9	185	┤
F BD0	A	122	z	F CE0	9A	154	╰	F DD0	BA	186	=
8	B	123	{	8	9B	155	╯	8	BB	187	┬
F BE0	C	124	¦	F CD0	9C	156	└	F DE0	BC	188	┌
8	D	125	}	8	9D	157	┘	8	BD	189	┐
F BF0	E	126	~	F CF0	9E	158	◺	F DF0	BE	190	└
8	F	127	▒	8	9F	159	◿	8	BF	191	┘

SORCERER SYSTEMS AND PERIPHERALS AVAILABLE FROM COMPUTER MART, INC.

8K Sorcerer	$895.00
16K Sorcerer	1150.00
32K Sorcerer	1395.00
S-100 Expansion Box	299.00
Leedex Video 100 12" Monitor	149.00
Integral Data Systems IP-125 Printer	799.00
Integral Data Systems IP-225 Printer	949.00
Integral Data Systems 440 Printer	995.00
Exidy S-100 Expansion Kit	149.00
16K Memory Add-On Kit	119.00
Installed and Guaranteed	199.00
Micropolis Mod I Disk Drive	795.00
Additional Drive	545.00
Micropolis Mod II Disk Drive	1145.00
Dual Drive Mod II	1895.00
Sorcerer Cable for IDS Printers	50.00
Sorcerer Operations Manual	9.95
Sorcerer BASIC Manual	11.95
Sorcerer Technical Manual	11.95
Sorcerer Standard BASIC ROM Pac	99.00
Sorcerer EPROM Pac	49.00
Serial Data Cassette Cable	75.00
D-30 Cassettes	1.50
Tek-Com Accoustic Coupler Modems	195.00
Cassette Recorders	55.00
Vista Disk Drives	595.00
Thinkertoys 8" Double Density Drives	1149.00
CP/M for Sorcerer	149.00
Full Line of S-100 Boards	
RMP Industries Music Box for Sorcerer	49.95
Sorcerer Software	
GRT "Clinic" and "Beat the House"	14.95
Creative Computing "Graphic Games-2"	7.95
Basically Speaking Software	
CP/M for Micropolis	145.00
Assorted Cables	
Other Printers	

"The Exidy Monitor"
back-issues

VOLUME I:

[ ] Issue 1: printer control; single-stroke BASIC; single character input routine; PEEK and POKE addresses
COST: $0.25

[ ] Issue 2: PEEK and POKE; the USR function; MEMTEST; memory chart; software library
COST: $1.00

[ ] Issue 3: the DEVELOPMENT PAC; questionnaires; Misc. info; Z-80 Machine language part 1; MUSICAL WALLPAPER
COST: $1.00

SOFTWARE VOLUMES

Volume	Contents	Non-Member price	Member price
VOL I:	AMAZE, LUNAR, ROTE	[ ] $9.99	[ ] $6.50
VOL II:	LUCAS, RSP, CHECK	[ ] $9.99	[ ] $6.50
VOL III:	DEPTH, REVRS, HIQUE	[ ] $9.99	[ ] $6.50
VOL IV:	BIO, MIDAG, MANDL	[ ] $9.99	[ ] $6.50
VOL V:	HANGM, HPAWN	[ ] $9.99	[ ] $6.50
VOL VI:	HANG2, HANG3	[ ] $9.99	[ ] $6.50
VOL VII:	WARPX, TREK1	[ ] $9.99	[ ] $6.50
VOL VIII:	WARPZ, TREK1	[ ] $9.99	[ ] $6.50
VOL IX:	BANER, BOXOP	[ ] $9.99	[ ] $6.50
VOL X:	DEMO, CHAR	[ ] $9.99	[ ] $6.50
VOL XI:	DUMBT, DUMB2	[ ] $9.99	[ ] $6.50

(enter amount wanted of each in backets)

NAME: ____________________
ADDRESS: ____________________
CITY: ____________ STATE: ________ ZIP: ____________

TOTAL PURCHASE: $______
Plus Shipping: $ 1.00
MA residents add 5% sales tax: ______

TOTAL: $______

Enclosed is $_____ for the above checked off volumes and/or back-issues.

[] CASH, CHECK, or MONEY ORDER
[] VISA/BANK AMERICARD [] MASTERCHARGE
Card #______________ Exp. date: ____________

Please allow 2 to 4 weeks for delivery